

The one page guide to L^AT_EX

W. Ethan Duckworth, Loyola College, Fall 2008

Figure 1: Sample L^AT_EX input

	1	<code>\documentclass{article}</code>
Loads extra fonts package.→	2	<code>\usepackage{amsfonts}</code>
Defines the command <code>\Z</code> →	3	<code>\newcommand{\Z}{\mathbb{Z}}</code>
which equals <code>\mathbb{Z}</code> .	4	<code>\newcommand{\Q}{\mathbb{Q}}</code>
	5	<code>\begin{document}</code>
Don't indent next word.→	6	<code>\noindent</code>
Force line to end with <code>\\</code> .→	7	John Doe\\
	8	Calculus I\\
	9	Homework 4
Adds extra space between paragraphs.→	10	<code>\bigskip</code>
Blank line starts new paragraph.→	11	
Math stuff in <code>\$. . .\$</code> will be text→	12	In this problem we will show that $\mathbb{Z} \subseteq \mathbb{Q}$. We start with the
style (in the line of text).	13	definition
Math stuff in <code>\$\$. . . \$\$</code> will be→	14	<code>\$\$</code>
display style (centered, on a	15	<code>\Q = \left\{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\right\}</code> .
line by itself, larger symbols).	16	<code>\$\$</code>
	17	Of course we assume that the reader is familiar with this.
Blank line starts new paragraph.→	18	
"Now" is indented in output.→	19	Now, we note that for each $a \in \mathbb{Z}$ we have that a equals $\frac{a}{1}$
Fractions in text style are smaller→	20	a . Since $\frac{a}{1} \in \mathbb{Q}$, this proves our claim.
than in display style (line 15).	21	<code>\end{document}</code>

Figure 2: Sample L^AT_EX output

Here's where `\bigskip` space goes.→
Note that the first line is indented here,→
and that L^AT_EX automatically wraps
the words.

Here's the display style math.→

Again, first line indented, automatic→
word wrapping. Also, note the small
fractions, as compared to the one in
the display style math.

John Doe
Calculus I
Homework 4

In this problem we will show that $\mathbb{Z} \subseteq \mathbb{Q}$. We start with the definition

$$\mathbb{Q} = \left\{ \frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0 \right\}.$$

Of course we assume that the reader is familiar with this.

Now, we note that for each $a \in \mathbb{Z}$ we have that a equals $\frac{a}{1}$. Since $\frac{a}{1} \in \mathbb{Q}$, this proves our claim.

1

Turn the page for some discussion of this example and L^AT_EX.

1. To use L^AT_EX you first create an input file. This file contains only plain text and has all of your text, math and commands in it. Then you run the program L^AT_EX, which takes your input file, and produces output, typically a PDF file (for Adobe Reader).
2. Here is a minimal example of a L^AT_EX input file

```
\documentclass{article}
\begin{document}
  Hello world.
\end{document}
```

The output would be something like this

```
Hello world.
```

1

3. L^AT_EX can make all sorts of nice math symbols, but you have to tell it where you want the math. Math stuff should be contained in a single pair of dollar signs “ \dots ” or a double pair of dollar signs “ \dots ”.

A single pair of dollar signs keeps the math stuff in the same line as the text. Figure 1 lines 12, 19, and 20 show input with math in a single pair. Figure 2 shows the output.

A double pair of dollar signs puts the math stuff centered, on a line by itself. This is called *display style* math, and it also makes things like fractions bigger, as well as summation symbols, limits, etc. Figure 1 line 15 shows input with math in a double pair. When I use double dollar signs, I usually enter them by themselves, as in lines 14 and 16. L^AT_EX doesn’t care that I enter them this way, and it makes no change in the output, it just helps me see the math stuff that will be in display style.

4. You can force L^AT_EX to start a new line by entering a blank line (i.e. hitting return twice) or by entering a double backslash `\`. Figure 1 lines 7,8 show an example of using `\` to force a new line to start. Line 18 shows an example of using a blank line.

There is a difference between these two ways of ending the line. Using `\` forces a new line to be started, but not a new paragraph. Putting a blank line in by hitting return twice actually starts a new paragraph. By default, L^AT_EX indents most new

paragraphs, so using a blank line will usually cause the next word to be indented (see the output from lines 12 and 19). Similarly, L^AT_EX views “John Doe” in line 7 as the beginning of the first paragraph, but the `\noindent` command in line 6 prevents this from being indented.

5. When you enter text, most of the time you should not try to force lines to end where you want, but instead let L^AT_EX wrap the words around.

To see how this works, compare Figure 1 lines 12, 17, 19 with the output. This automatic wrapping works perfectly with words and sentences. It doesn’t work out as well with the way people sometimes write homework or notes where lots of lines are stacked one above the other in a way that is supposed to line up.

If you want to line up equations and symbols in successive lines, you should read about `array`, `eqnarray`, `align`, etc. in a longer guide than this one. However, I think that a lot of the time the best solution is to use a few more words and sentences, and give L^AT_EX room to do its wrapping (I believe this actually produces better written mathematics).

6. One nice feature about L^AT_EX is that you can load extra packages with specialized capabilities, and you can define your own commands that are easy to read, or that do fancy things.

Figure 1 line 2, shows you how to load the `amsfonts` package. This package is created by the AMS (American Mathematical Society) and defines some extra fonts. Line 3 shows how to create a new command named `\Z`. This command is defined to mean the same thing as `\mathbb{Z}`. The package `amsfonts` defines the command `\mathbb` to produce the special fonts called Blackboard bold. Lines 12, 15, 19, 20 use the commands `\Z` and `\Q`, and Figure 2 shows the output.

7. It is possible to force in extra space between paragraphs, or for indenting things, etc. However, as in number 5 above, it is usually best to let L^AT_EX take care of these things automatically. Occasionally, you might want to insert a little extra space. Figure 1 line 10 shows how to use `\bigskip` to do this. Note that there is a blank line *after* `\bigskip`, and this is where the extra space goes.